

# Manual de Uso de Optuna

-Cristhian Arlindo Mamani Nina  
-Edison Antony Sayritupa Coaricona

February 13, 2025

## ¿Qué es Optuna?

Optuna es una biblioteca de optimización de hiperparámetros automatizada, diseñada para mejorar el rendimiento de los modelos de *machine learning*. Se utiliza para encontrar los mejores valores de los hiperparámetros de un modelo a través de un proceso de optimización. El objetivo es encontrar la combinación óptima de parámetros que maximice o minimice una función objetivo, generalmente asociada con el rendimiento del modelo, como la precisión, la pérdida, entre otros.

La optimización de hiperparámetros es un paso crucial en la construcción de modelos de *machine learning*, ya que el ajuste de los parámetros adecuados puede mejorar significativamente el rendimiento del modelo. Sin embargo, explorar todas las posibles combinaciones de parámetros es una tarea computacionalmente costosa. Ahí es donde Optuna entra en juego, proporcionando una solución eficiente para este problema.

## ¿Cómo funciona Optuna?

Optuna utiliza un enfoque basado en estudios (*studies*) y pruebas (*trials*) para optimizar los hiperparámetros. A continuación, se describe el proceso básico de optimización en Optuna:

- **Estudio:** Un estudio es un conjunto de optimización en el que se buscan los mejores valores para los hiperparámetros. Un estudio contiene múltiples pruebas que evalúan distintas configuraciones de parámetros.
- **Prueba (Trial):** Una prueba es una evaluación individual de un conjunto específico de hiperparámetros. Optuna genera diferentes combi-

naciones de valores para los hiperparámetros y las evalúa mediante la función objetivo.

- **Espacio de búsqueda:** El espacio de búsqueda define los posibles valores que pueden tomar los hiperparámetros. Por ejemplo, se puede definir un rango de números flotantes, valores enteros o incluso seleccionar entre un conjunto de opciones discretas.
- **Función objetivo:** La función objetivo es la función que se intenta optimizar, que generalmente mide el rendimiento del modelo en función de los parámetros seleccionados. La optimización de esta función es el propósito central de Optuna.

Optuna utiliza técnicas avanzadas de optimización para buscar en el espacio de hiperparámetros de manera eficiente, sin necesidad de explorar todas las combinaciones posibles. Entre los algoritmos más utilizados se encuentran:

- **Tree-structured Parzen Estimator (TPE):** Un algoritmo basado en modelos probabilísticos que es particularmente eficaz para problemas no lineales y complejos.
- **Optimización basada en bandit:** Utiliza estrategias de optimización tipo "multi-armed bandit", que ajustan el enfoque según los resultados obtenidos.

## Ventajas de Optuna

Optuna ofrece varias ventajas que la hacen una herramienta poderosa para la optimización de modelos de *machine learning*:

- **Exploración eficiente:** Utiliza algoritmos avanzados para encontrar la mejor combinación de parámetros con menos pruebas.
- **Facilidad de uso:** Su API sencilla permite integrarlo fácilmente con frameworks como TensorFlow, Keras, PyTorch y Scikit-learn.
- **Escalabilidad:** Es compatible con entornos distribuidos y en la nube, adecuado para proyectos a gran escala.
- **Variedad de algoritmos:** Soporta múltiples algoritmos de optimización, como *TPE*, *random search* y *grid search*.
- **Optimización condicional:** Permite optimizar hiperparámetros de manera condicional, según otras selecciones previas.

## Acceder a la Página Oficial de Optuna

1. Abre tu navegador web y dirígete a la URL: <https://optuna.org>.
2. En la página de inicio encontrarás una descripción sobre qué es Optuna y cómo ayuda a optimizar modelos de Machine Learning. Asegúrate de que la barra de direcciones esté visible en tu captura de pantalla.



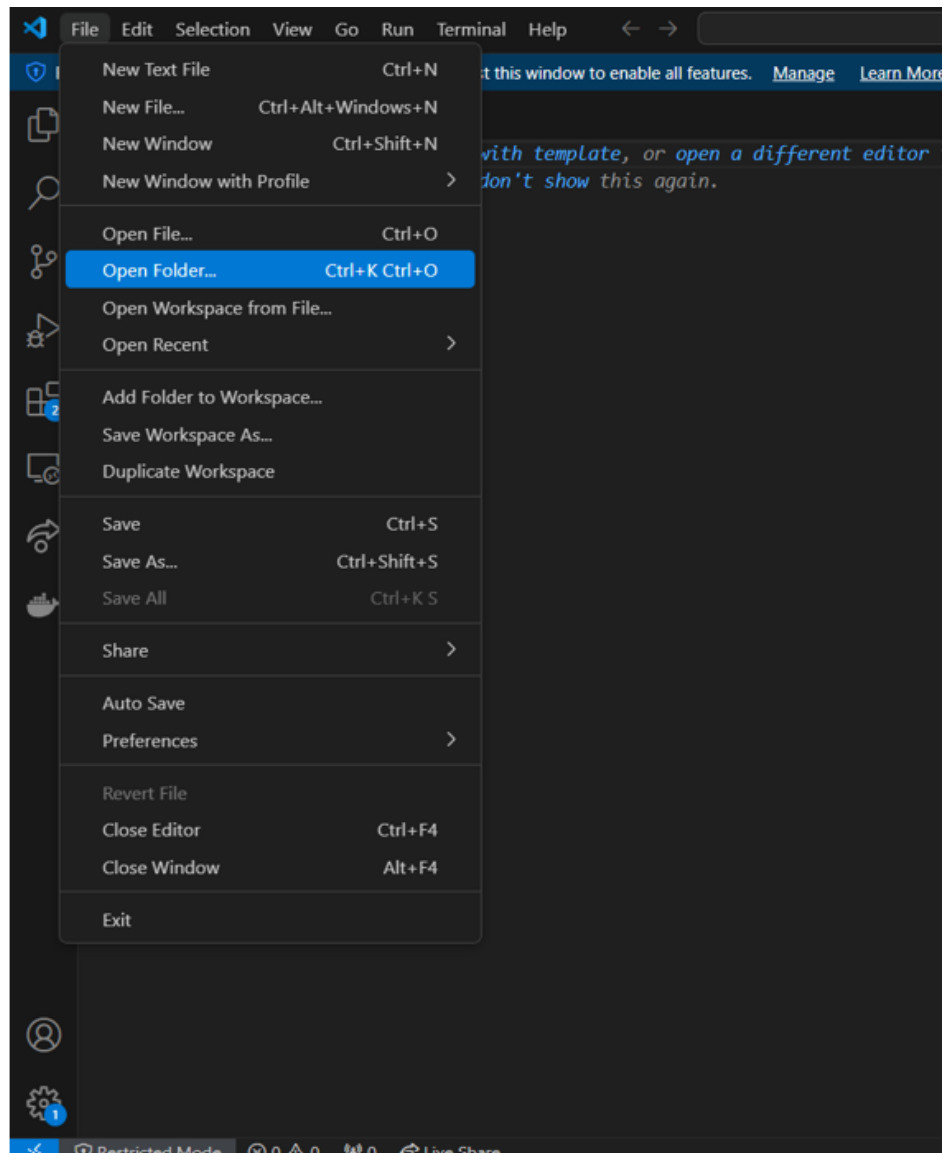
Figure 1: Captura de la página de inicio de Optuna

## Instalación y Configuración

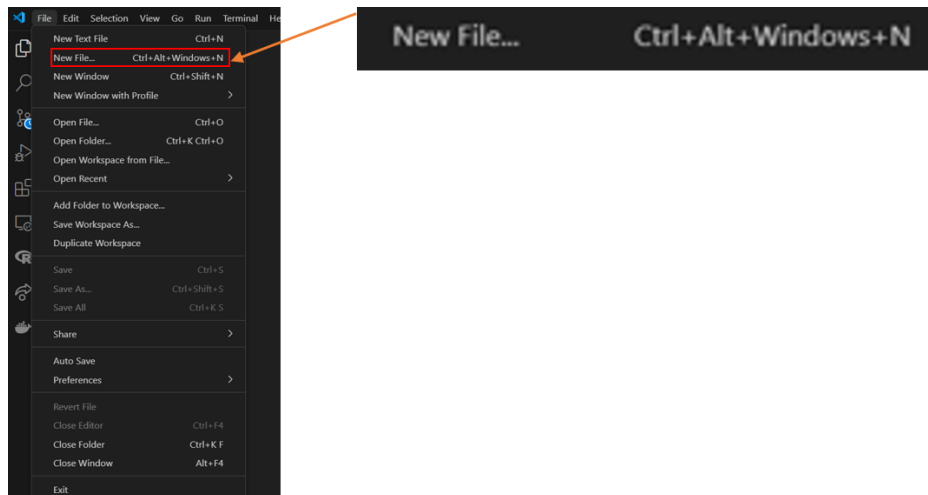
### Instalar Optuna

Crear una nueva carpeta:

-Abre VSCode en tu computadora.

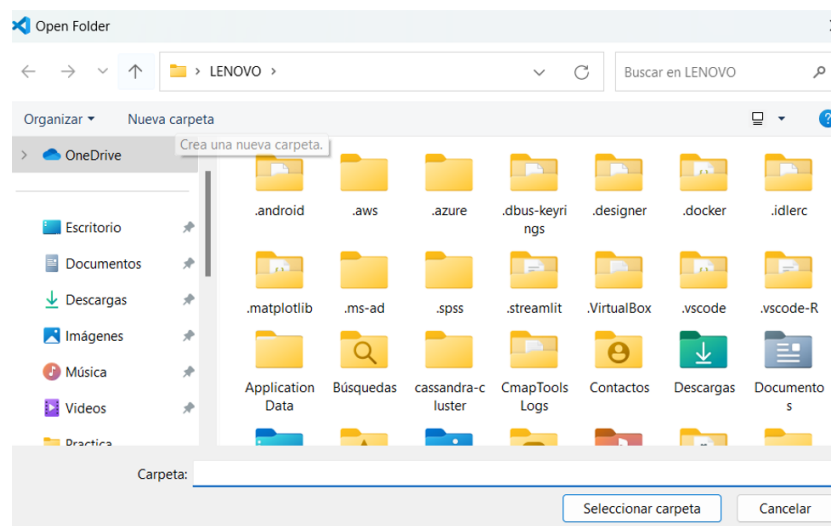


-Abre tu terminal o entorno de desarrollo (por ejemplo, Jupyter Notebook o VSCoDe).

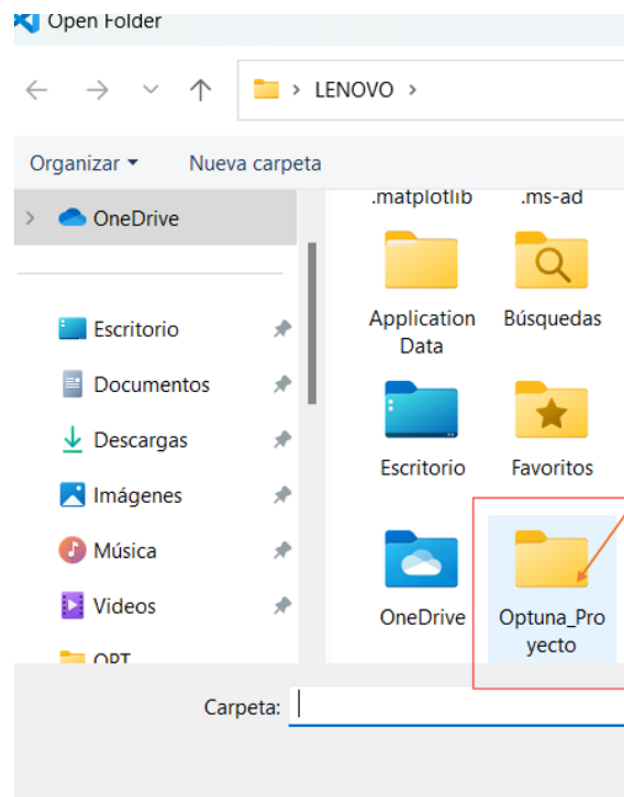


## Crea una nueva carpeta para tu proyecto de Optuna.

-Si estás en Windows, haz clic derecho en tu escritorio o en la ubicación deseada y selecciona "Nuevo" y luego "Carpeta".

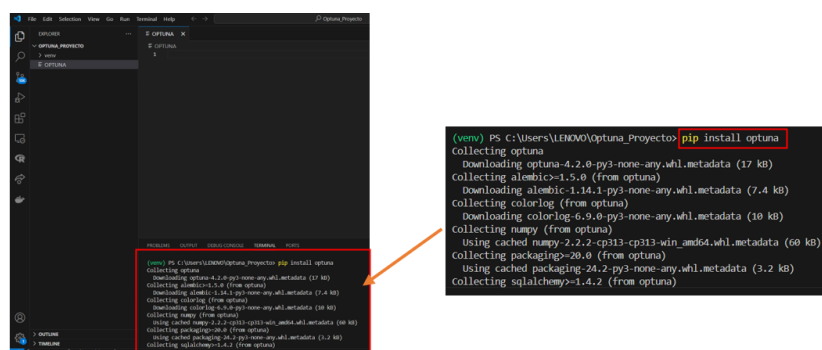


-Nombrala de acuerdo a tu proyecto, por ejemplo, *Optuna\_proyecto*.



-Ejecuta el siguiente comando para instalar Optuna:

```
pip install optuna
```



-Si ves un error relacionado con Scikit-Learn, instálalo con el siguiente comando:

```
pip install scikit-learn
```

```
pip install scikit-learn

Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Using cached threadpoolctl-3.5.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.6.1-cp313-cp313-win_amd64.whl (11.1 MB)
  11.1/11.1 MB 8.7 MB/s eta 0:00:00
Using cached joblib-1.4.2-py3-none-any.whl (301 kB)
Downloading scipy-1.15.1-cp313-cp313-win_amd64.whl (43.6 MB)
  43.6/43.6 MB 11.2 MB/s eta 0:00:00
Using cached threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.4.2 scikit-learn-1.6.1 scipy-1.15.1 threadpoolctl-3.5.0
```

-Verificación de la instalación:

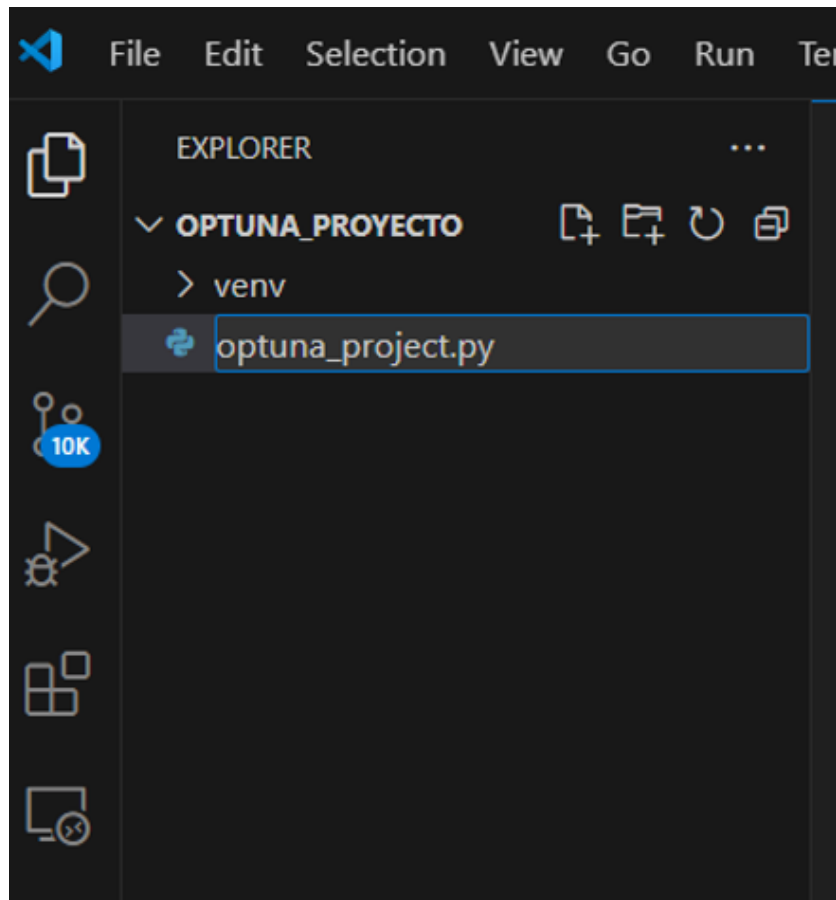
```
python -c "import sklearn; print('Scikit-Learn instalado correctamente')"
```

```
>>> python -c "import sklearn; print('Scikit-Learn instalado correctamente')"
Scikit-Learn instalado correctamente
(venv) PS C:\Users\LENOVO\Optuna_Proyecto>
```

Figure 2: Instalación de Optuna

## Crear un Proyecto en VSCode

1. Crea una carpeta para tu proyecto de Optuna.
2. Abre VSCode y selecciona "Abrir carpeta..." en el menú File.



3. Crea un entorno virtual en la terminal de VSCode con el comando:

```
python -m venv venv
```

4. Activa el entorno virtual:

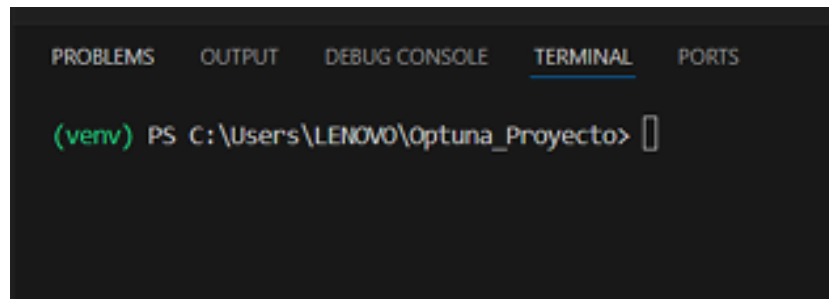
- **Windows:**

```
.\venv\Scripts\activate
```

- **macOS/Linux:**

```
source venv/bin/activate
```





## Escribir el Código Inicial en Python

1. Crea un archivo llamado `optuna_project.py` y agrega el siguiente código de prueba:

```
import optuna

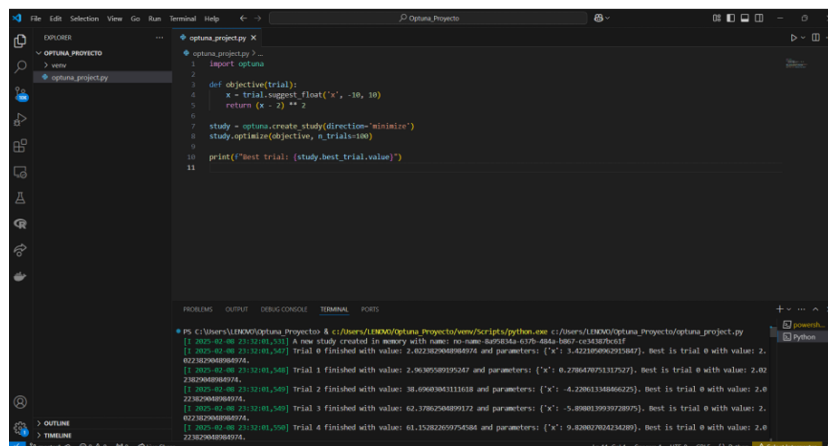
def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)

print(f"Best trial: {study.best_trial.value}")
```

2. Ejecuta el archivo:

```
python optuna_project.py
```



```
1 import optuna
2
3 def objective(trial):
4     x = trial.suggest_float("x", -10, 10)
5     return (x - 2) ** 2
6
7 study = optuna.create_study(direction="minimize")
8 study.optimize(objective, n_trials=100)
9
10 print("Best trial: (study.best_trial.value)")
11
```

```
[1] 2025-02-08 21:31:01,511 A new study created in memory with name: no-name-8a533da-437b-48da-b8d7-c2343276c31f
[1] 2025-02-08 21:31:01,547 Trial 0 finished with value: 2.0232060868974 and parameters: {"x": 1.422399962915847}. Best is trial 0 with value: 2.0232060868974.
[1] 2025-02-08 21:31:01,549 Trial 1 finished with value: 2.9639589195247 and parameters: {"x": 0.37864707131727}. Best is trial 0 with value: 2.0232060868974.
[1] 2025-02-08 21:31:01,549 Trial 2 finished with value: 38.6960384111618 and parameters: {"x": -4.22061348666223}. Best is trial 0 with value: 2.0232060868974.
[1] 2025-02-08 21:31:01,549 Trial 3 finished with value: 62.3786240899172 and parameters: {"x": -5.8988139939728975}. Best is trial 0 with value: 2.0232060868974.
[1] 2025-02-08 21:31:01,550 Trial 4 finished with value: 61.152822659754584 and parameters: {"x": 9.820807824234289}. Best is trial 0 with value: 2.0232060868974.
```

Figure 3: Código de prueba de Optuna en VSCode

## Descripción del Dataset

El archivo **Registro Nacional de Infractores\_00.csv** contiene información detallada sobre infractores registrados en un sistema de control y sanción. Este dataset se utiliza para el análisis de datos y el entrenamiento de modelos de aprendizaje automático con el objetivo de predecir patrones de sanciones y mejorar la clasificación de infractores.

## Características del Dataset

- **Formato:** CSV (Comma-Separated Values)
- **Estructura:** Contiene múltiples columnas con datos sobre los infractores, sus sanciones y otros atributos relevantes.
- **Contenido:**
  - \* **Órgano Sancionador:** Entidad responsable de la sanción.
  - \* **Fecha Resolución:** Fecha en la que se emitió la sanción.
  - \* **Otros Atributos:** Variables adicionales relacionadas con el infractor y la sanción.

## Objetivo del Uso

Este dataset es utilizado para los siguientes propósitos:

- **Limpieza de Datos:** Eliminación de valores nulos y transformación de datos categóricos en numéricos.
- **Optimización de Modelos:** Aplicación de técnicas de aprendizaje automático como *Random Forest* y *SVM*.
- **Análisis Exploratorio:** Identificación de patrones en las resoluciones de sanciones.
- **Predicción de Sanciones:** Implementación de modelos predictivos para clasificar posibles infractores.

Enlace

{[https://github.com/cristhian-arlindo16/GUIA\\_OPTUNA/blob/main/Registro](https://github.com/cristhian-arlindo16/GUIA_OPTUNA/blob/main/Registro)}

## 1. Cómo Cambiar los Datos: Un Ejemplo

### Eliminar Valores Nulos

En ocasiones, los datos contienen valores nulos o vacíos. Para eliminarlos, puedes usar el siguiente código:

```
df.dropna(inplace=True)
```

Este código eliminará todas las filas que contengan valores nulos.



```

1 Bob NaN Los Angeles
PS C:\Users\LENOVO\Optuna_Proyecto> ^C
PS C:\Users\LENOVO\Optuna_Proyecto> & c:/Users/LENOVO/Optuna_Proyecto/venv/Scripts/python.exe c:/Users/LENOVO/Optuna_Proyecto/ejempl1.py
Dataframe original con valores nulos:
  Nombre  Edad  Ciudad
0  Alice  25.0  New York
1  Bob    NaN  Los Angeles
2  Charlie 30.0  Chicago
3  NaN    22.0  Miami
4  Eve    NaN  Dallas

Dataframe después de eliminar valores nulos:
  Nombre  Edad  Ciudad
0  Alice  25.0  New York
2  Charlie 30.0  Chicago

El gráfico se ha guardado como 'grafico_eliminacion_nulos.png'
PS C:\Users\LENOVO\Optuna_Proyecto>

```

Figure 4: Ejemplo de eliminación de valores nulos en el dataset.

Puedes ver el ejemplo completo en el siguiente enlace de GitHub:  
[https://github.com/cristhian-arlindo16/GUIA\\_OPTUNA/blob/main/Ejemplo1](https://github.com/cristhian-arlindo16/GUIA_OPTUNA/blob/main/Ejemplo1)

## Codificar Variables Categóricas

Si tienes columnas con valores categóricos (por ejemplo, "Órgano Sancionador"), puedes convertirlas en números utilizando el codificador `LabelEncoder`:

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
df["Órgano Sancionador"] = encoder.fit_transform(df["Órgano Sancionador"])
```

```
Archivo cargado correctamente.
Columnas en el DataFrame: Index(['Número', 'Infractor', 'Documento de Identidad', 'Título habilitante',
'Resolución sancionadora', 'Fecha Resolución', 'Órgano Sancionador',
'Ámbito de la infracción'],
dtype=object)
Codificación completada para 'Órgano Sancionador'.
Número Infractor Documento de Identidad ... Fecha Resolución Órgano Sancionador Ámbito de la Infracción
0 1 ABANO VEINTEMILLA JONATHAN 4963135 ... 13/09/2016 2 GORE Madre de Dios
1 2 ABANO VEINTEMILLAS ANGEL DANIEL 5061884 ... 31/08/2015 0 GORE Madre de Dios
2 3 ABANTO HUACCHA ELEUTERIO OLEGARIO 67690 ... 4/12/2014 0 GORE Ucayali
3 4 ABANTO VARGAS MARIA DEL CARMEN 27906159 ... 11/08/2015 2 ATFFS Cajamarca
4 5 ABENSUR REATEGUI LUZ PATRICIA MARTINA 47478469 ... 28/04/2015 0 GORE Huánuco

[5 rows x 8 columns]
PS C:\Users\LENOVO\Optuna_Proyecto> []
```

Figure 5: Ejemplo de eliminación de valores nulos en el dataset.

Puedes ver el ejemplo completo en el siguiente enlace de GitHub:  
[https://github.com/cristhian-arlindo16/GUIA\\_OPTUNA/blob/main/Ejemplo1](https://github.com/cristhian-arlindo16/GUIA_OPTUNA/blob/main/Ejemplo1)

## Convertir Fechas a Años

Para convertir fechas a un formato que sea comprensible para el modelo, puedes extraer solo el año de una columna de fechas:

```
df["Fecha Resolución"] = pd.to_datetime(df["Fecha Resolución"], dayfirst=True)
```

```
PS C:\Users\LENOVO\Optuna_Proyecto> ^C
PS C:\Users\LENOVO\Optuna_Proyecto> & c:/Users/LENOVO/venv/Scripts/python.exe c:/Users/LENOVO/Optuna_Proyecto/ejemplo2.py
Número Infractor Documento de Identidad ... Órgano Sancionador Ámbito de la Infracción Año Resolución
0 1 ABANO VEINTEMILLA JONATHAN 4963135 ... OSINFOR GORE Madre de Dios 2016.0
1 2 ABANO VEINTEMILLAS ANGEL DANIEL 5061884 ... ARFFS GORE Madre de Dios 2015.0
2 3 ABANTO HUACCHA ELEUTERIO OLEGARIO 67690 ... ARFFS GORE Ucayali 2014.0
3 4 ABANTO VARGAS MARIA DEL CARMEN 27906159 ... OSINFOR ATFFS Cajamarca 2015.0
4 5 ABENSUR REATEGUI LUZ PATRICIA MARTINA 47478469 ... ARFFS GORE Huánuco 2015.0

[5 rows x 9 columns]
PS C:\Users\LENOVO\Optuna_Proyecto> []
```

Figure 6: Conversión de fechas a formato de año.

Puedes ver el ejemplo completo en el siguiente enlace de GitHub:  
[https://github.com/cristhian-arlindo16/GUIA\\_OPTUNA/blob/main/Ejemplo1](https://github.com/cristhian-arlindo16/GUIA_OPTUNA/blob/main/Ejemplo1)

## 2. Código y Interpretación de los Resultados de Optuna

Cuando usas Optuna para optimizar un modelo, obtienes dos tipos de resultados importantes:

1. **Precisión (accuracy)** del modelo.
2. **Hiperparámetros optimizados** que Optuna encontró.

Configuración de estilos para código Python:

### Cargar y Preparar el Dataset

El dataset se carga desde un archivo CSV llamado **Registro Nacional de Infractores\_0.csv**. Se realizan los siguientes pasos:

- Eliminación de valores nulos.
- Conversión de fechas a formato de años.
- Codificación de variables categóricas con **LabelEncoder**.
- Selección de columnas numéricas para el modelo.

Código en Python:

```
import optuna
import matplotlib
matplotlib.use('Agg') # Cambiar el backend a 'Agg'
                        para evitar el uso de Tkinter
import pandas as pd
import numpy as np
import sklearn.ensemble
import sklearn.model_selection
import matplotlib.pyplot as plt

# 1. Cargar y preparar el dataset
def load_infractores():
    """Carga el dataset 'Registro Nacional de
        Infractores_0.csv' en un formato adecuado
        para ML"""

    # Cargar el CSV
```

```

df = pd.read_csv("Registro Nacional de
Infractores_0.csv", sep=None, engine="python"
, on_bad_lines="skip")

# Eliminar valores nulos
df.dropna(inplace=True)

# Convertir fechas a años si existe la columna
"Fecha Resoluci n"
if "Fecha Resoluci n" in df.columns:
    df["Fecha Resoluci n"] = pd.to_datetime(df[
"Fecha Resoluci n"], dayfirst=True,
errors='coerce').dt.year

# Codificar variables categóricas
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for col in df.select_dtypes(include=["object"]).
columns:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(
df[col])

# Seleccionar solo columnas numéricas
df_numeric = df.select_dtypes(include=[np.number
])

# Definir la variable objetivo
target_column = "Fecha Resoluci n" if "Fecha
Resoluci n" in df.columns else None
X = df_numeric.drop(columns=[target_column],
errors="ignore").values if target_column else
df_numeric.values
y = df[target_column].values if target_column
else None

return X, y

# 2. Definir la función objetivo para Optuna
def objective(trial):
    # Cargar el dataset
    X, y = load_infractores()

```

```

# Par metros a optimizar
n_estimators = trial.suggest_int("n_estimators",
    2, 20)
max_depth = int(trial.suggest_float("max_depth",
    1, 32, log=True))

clf = sklearn.ensemble.RandomForestClassifier(
    n_estimators=n_estimators, max_depth=
    max_depth)

# Evaluar con validaci n cruzada
return sklearn.model_selection.cross_val_score(
    clf, X, y, n_jobs=-1, cv=3).mean()

# 3. Ejecutar la optimizaci n
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=100)

# 4. Mostrar los mejores resultados
trial = study.best_trial

print("Mejor Precisi n:", trial.value)
print("Mejores Hiperpar metros:", trial.params)

# 5. Generar gr fico de la evoluci n de la
    optimizaci n
fig = optuna.visualization.matplotlib.
    plot_optimization_history(study)

# Guardar el gr fico como imagen
fig.figure.savefig("optuna_resultados.png")

# 6. Cerrar la figura correctamente
plt.close(fig.figure) # Cerramos la figura asociada

```

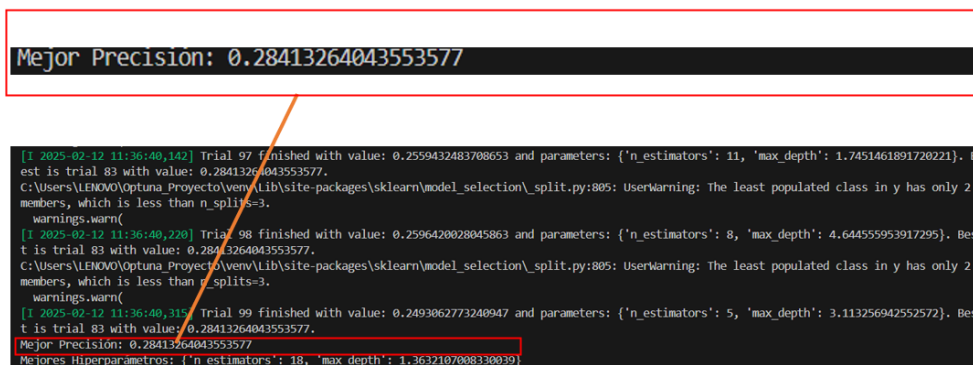
Puedes ver el ejemplo completo en el siguiente enlace de GitHub:  
[https://github.com/cristhian-arlindo16/GUIA\\_PTUNA/blob/main/Ejemplo1](https://github.com/cristhian-arlindo16/GUIA_PTUNA/blob/main/Ejemplo1)

## Precisión del Modelo

La precisión es una medida de qué tan bien el modelo predice los resultados. Puedes imprimir la precisión con el siguiente código:

```
print("Best Accuracy: {}".format(trial.value))
```

Por ejemplo, si la salida es 0.95, significa que el modelo tiene una precisión del 95%.



```
[I 2025-02-12 11:36:40,142] Trial 97 finished with value: 0.2559432483708653 and parameters: {'n_estimators': 11, 'max_depth': 1.7451461891720221}. Best is trial 83 with value: 0.28413264043553577.
c:\Users\LENOVO\Optuna Proyecto\venv\Lib\site-packages\sklearn\model_selection\_split.py:805: UserWarning: The least populated class in y has only 2 members, which is less than n_splits=3.
  warnings.warn(
[I 2025-02-12 11:36:40,220] Trial 98 finished with value: 0.2596420028045863 and parameters: {'n_estimators': 8, 'max_depth': 4.644555953917295}. Best is trial 83 with value: 0.28413264043553577.
c:\Users\LENOVO\Optuna Proyecto\venv\Lib\site-packages\sklearn\model_selection\_split.py:805: UserWarning: The least populated class in y has only 2 members, which is less than n_splits=3.
  warnings.warn(
[I 2025-02-12 11:36:40,315] Trial 99 finished with value: 0.2493062773240947 and parameters: {'n_estimators': 5, 'max_depth': 3.113256942552572}. Best is trial 83 with value: 0.28413264043553577.
Mejor Precisión: 0.28413264043553577
Mejores Hiperparámetros: {'n_estimators': 18, 'max_depth': 1.3632107008330039}
```

Figure 7: Gráfico de la precisión del modelo en función de los hiperparámetros.

## Hiperparámetros

Optuna también encuentra los mejores hiperparámetros para el modelo. Puedes imprimir los hiperparámetros óptimos de la siguiente manera:

```
print("Best Hyperparameters: {}".format(trial.params))
```

## 3. Interpretar los Valores de los Hiperparámetros

### n\_estimators

El parámetro `n_estimators` define el número de árboles en el modelo de Random Forest.



```
Mejores Hiperparámetros: {'n_estimators': 18, 'max_depth': 1.3632107008330039}

[I 2025-02-12 11:36:40,142] Trial 97 finished with value: 0.2559432483708653 and parameters: {'n_estimators': 11, 'max_depth': 1.7451461891720221}. Best is trial 83 with value: 0.28413264043553577.
C:\Users\LENOVO\Optuna_Proyecto\venv\Lib\site-packages\sklearn\model_selection\_split.py:805: UserWarning: The least populated class in y has only 2 members, which is less than n_splits=3.
  warnings.warn(
[I 2025-02-12 11:36:40,220] Trial 98 finished with value: 0.2596420028045863 and parameters: {'n_estimators': 8, 'max_depth': 4.644555953917295}. Best is trial 83 with value: 0.28413264043553577.
C:\Users\LENOVO\Optuna_Proyecto\venv\Lib\site-packages\sklearn\model_selection\_split.py:805: UserWarning: The least populated class in y has only 2 members, which is less than n_splits=3.
  warnings.warn(
[I 2025-02-12 11:36:40,315] Trial 99 finished with value: 0.2493062773240947 and parameters: {'n_estimators': 5, 'max_depth': 3.113256942552572}. Best is trial 83 with value: 0.28413264043553577.
Mejor Precisión: 0.28413264043553577
Mejores Hiperparámetros: {'n_estimators': 18, 'max_depth': 1.3632107008330039}
```

Figure 8: Gráfico mostrando los mejores hiperparámetros encontrados por Optuna.

- **Mejor valor:** 18
- **Precisión:** 0.2841326404355377
- **Tiempo de entrenamiento:** Aumenta con el valor de `n_estimators`.

Best Hyperparameters: {'n\_estimators': 18, 'max\_depth': 1.3632}

## max\_depth

El parámetro `max_depth` controla la profundidad máxima de los árboles.

- **Mejor valor:** 1.3632
- **Precisión:** 0.2841326404355377
- **Riesgo de sobreajuste:** Controlado por la profundidad moderada.

Best Hyperparameters: {'n\_estimators': 18, 'max\_depth': 1.3632}

## svc\_c en SVM

El parámetro `svc_c` regula el ajuste entre un modelo más general y uno más ajustado.

- **Valor de C:** 1.0 (ajuste moderado)

- **Flexibilidad del modelo:** Equilibrada entre precisión y generalización.

```
clf = sklearn.svm.SVC(C=1.0, gamma="auto")
```

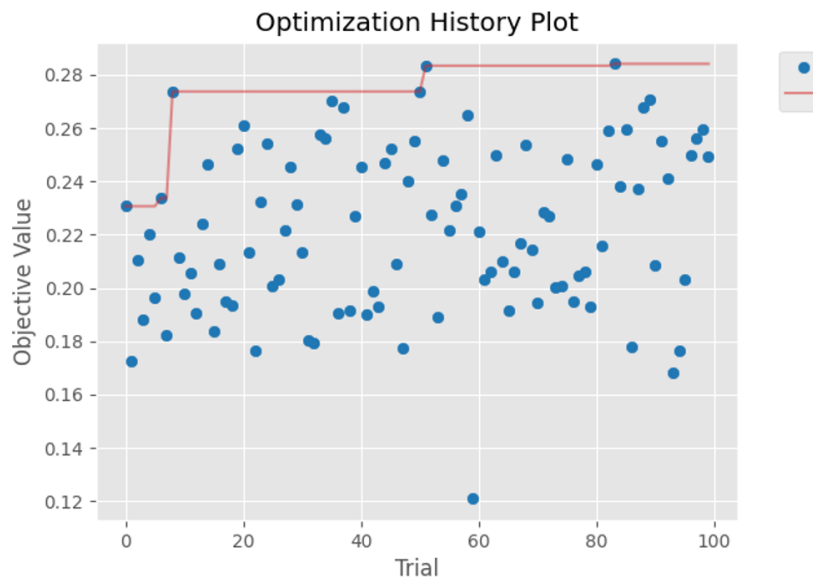


Figure 9: Gráfico que muestra la interpretación de los hiperparámetros en el modelo. En este gráfico, la relación entre `n_estimators` y `max_depth` es visualizada, mostrando cómo afectan la precisión del modelo.

## Resultados de la Optimización

Durante la optimización del modelo con Optuna, se registraron los siguientes resultados en términos de la precisión obtenida y los hiperparámetros seleccionados:

```
[I 2025-02-12 20:41:39,663] A new study created in
memory with name: no-name-37e17e78-f7d2-43dd-bee1
-4ed6453232bf
[I 2025-02-12 20:41:42,359] Trial 0 finished with
value: 0.18989359069537246 and parameters: {'
n_estimators': 4, 'max_depth':
6.198060742610558}. Best is trial 0 with value:
0.18989359069537246.
```

```

[I 2025-02-12 20:41:44,448] Trial 1 finished with
value: 0.23293244246473643 and parameters: {'
n_estimators': 17, 'max_depth':
3.8639325830082134}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,508] Trial 2 finished with
value: 0.22696857213560998 and parameters: {'
n_estimators': 12, 'max_depth':
25.122446434381704}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,602] Trial 3 finished with
value: 0.20100140229316177 and parameters: {'
n_estimators': 18, 'max_depth':
2.6618087707081006}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,685] Trial 4 finished with
value: 0.15056504165635567 and parameters: {'
n_estimators': 3, 'max_depth':
2.4951846717151285}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,795] Trial 5 finished with
value: 0.22401055844262974 and parameters: {'
n_estimators': 12, 'max_depth':
11.912095349743602}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,874] Trial 6 finished with
value: 0.2069438257856966 and parameters: {'
n_estimators': 11, 'max_depth':
7.61192982231948}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:46,952] Trial 7 finished with
value: 0.2158690093211251 and parameters: {'
n_estimators': 14, 'max_depth':
3.8591747871334525}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:47,044] Trial 8 finished with
value: 0.21956446424152434 and parameters: {'
n_estimators': 14, 'max_depth':
16.10566294122866}. Best is trial 1 with value:
0.23293244246473643.
[I 2025-02-12 20:41:47,104] Trial 9 finished with
value: 0.24554813165058154 and parameters: {'
n_estimators': 9, 'max_depth':

```

```
4.816368046889566}. Best is trial 9 with value:  
0.24554813165058154.  
[I 2025-02-12 20:41:47,175] Trial 10 finished with  
value: 0.22696857213560998 and parameters: {'  
n_estimators': 8, 'max_depth':  
1.1981159075395305}. Best is trial 9 with value:  
0.24554813165058154.  
[I 2025-02-12 20:41:47,256] Trial 11 finished with  
value: 0.26632846655118375 and parameters: {'  
n_estimators': 20, 'max_depth':  
1.7323302948349852}. Best is trial 11 with value:  
0.26632846655118375.
```



Figure 10: link de codigos y csv el datased

URL

[https://github.com/cristhian-arlindo16/GUIA\\_PTUNA](https://github.com/cristhian-arlindo16/GUIA_PTUNA)